**Mô Hình Dự Đoán Dịch Vụ Sự Cố**

**1. Giới Thiệu Về Bộ Dataset**

**1.1. Số Lượng Hàng và Cột**

* Bộ dữ liệu hiện tại có tổng cộng 450671 hàng và 8 cột.
* Mỗi hàng trong bộ dữ liệu đại diện cho một quan sát tại một thời điểm cụ thể về một dịch vụ.

**1.2. Ý Nghĩa Của Mỗi Hàng/Cột**

Bộ dữ liệu chứa các đặc trưng chính sau đây:

* Day: Ngày thu thập dữ liệu, kiểu dữ liệu là object (chuỗi ngày).
* DoWeek: Ngày trong tuần (có thể từ 0-6, với 0 là Chủ Nhật).
* hour: Giờ trong ngày, kiểu int.
* count: Tổng số yêu cầu mà dịch vụ nhận được tại thời điểm đó.
* passed: Số lượng yêu cầu được xử lý thành công.
* period: Thời gian xử lý trung bình.
* data: Thông tin dữ liệu hoặc giá trị đo lường tại thời điểm đó (có thể là mức sử dụng tài nguyên).
* ServiceID: Định danh của dịch vụ được giám sát.

**2. Giới Thiệu Về Vấn Đề Muốn Giải Quyết**

**a. Input là thời gian T**

* Bộ dữ liệu cần dự đoán dịch vụ nào có thể gặp lỗi tại thời điểm T+deltatime dựa trên các đặc trưng về thời gian, count, passed.

**b. Output**

* Là các dịch vụ bị lỗi tại thời điểm T+deltatime từ đó đưa ra cảnh báo để xử lý sớm.

**3. Giới Thiệu Về Cách Train/Test Split**

**a. Dữ liệu được thu từ khi nào tới khi nào**

* Bộ dữ liệu được thu thập từ 29/12/2021 đến 24/7/2023

**b. Dữ liệu train/test được lấy từ khi nào đến khi nào**

* **Train-set:** từ 29/12/2021 đến thời điểm T (được nhập từ bàn phím)
* **Test-set:** từ thời điểm T đến T + deltatime

**4. Ứng với một thời điểm T: Giới thiệu về observation trong khoảng thời gian từ T -> T + deltatime**

**a. Cách để lấy observation**

* Dữ liệu có thể được lấy bằng cách lựa chọn những hàng nằm trong khoảng T – (T + deltatime)

**b. Mỗi observation có bao nhiêu hàng và bao nhiêu cột**

* Một observation có 8 cột, bao gồm Day, DoWeek, hour, count, passed, period, data, ServiceID.

**c. Số cột có cố định không**

* Có, số cột là cố định (8 cột).

**d. Số hàng có cố định không**

* Không, số hàng tùy thuộc vào thời gian deltatime được chọn.

**e. Làm sao để chuyển các đặc trưng về dạng vector số:**

* Các cột thời gian như Day được chuyển đổi thành các thành phần như year, month, day, day\_of\_week.
* Các cột số thực như period, data được chuẩn hóa để đưa về cùng thang đo.

**5. Ứng với một thời điểm T: Giới thiệu về label**

**a. Làm thế nào để sử dụng multi-label**

Multi-label Classification là một loại bài toán phân loại mà trong đó mỗi mẫu có thể thuộc về nhiều nhãn khác nhau thay vì chỉ một nhãn duy nhất. Điều này đặc biệt quan trọng trong tình huống bạn muốn dự đoán nhiều dịch vụ có thể gặp lỗi cùng lúc.

* **Cách sử dụng multi-label trong dữ liệu hiện tại**:
* Để chuẩn bị dữ liệu cho mô hình multi-label, bạn cần chuyển cột ServiceID hiện tại sang một dạng nhãn phù hợp để biểu diễn nhiều nhãn. Thông thường, ta sử dụng một **ma trận nhãn** với các giá trị 0 hoặc 1, trong đó mỗi cột tương ứng với một dịch vụ, và mỗi dòng tương ứng với một mẫu.
* **Ví dụ về cách tạo nhãn multi-label**:
* Nếu chúng ta có 3 dịch vụ ServiceID\_2, ServiceID\_5, ServiceID\_7 và muốn biểu diễn các nhãn tại thời điểm T, một dòng trong ma trận nhãn có thể là [1, 0, 1, 0, 1], trong đó:

+ 1 là dấu hiệu dịch vụ có khả năng bị lỗi.

+ 0 là dấu hiệu dịch vụ hoạt động bình thường.

* **Kỹ thuật biểu diễn multi-label**:
* Trong **scikit-learn**, có một số phương pháp để xử lý dữ liệu dạng multi-label:
* **Binary Relevance**: Chuyển bài toán multi-label thành **n bài toán nhị phân** độc lập. Mỗi bài toán phân loại xem xét chỉ một nhãn.
* **Classifier Chains**: Chuỗi các bộ phân loại, trong đó mỗi bộ phân loại dựa trên các đầu ra của các bộ phân loại trước đó.
* **Label Powerset**: Chuyển đổi tất cả các nhãn thành một tổ hợp duy nhất và xử lý như bài toán phân loại đa lớp (multi-class).

**b. Giới thiệu NN network với sigmoid activation ở ngõ ra**

* Neural Network với hàm kích hoạt sigmoid ở đầu ra giúp mô hình dự đoán xác suất từng dịch vụ gặp lỗi.
* **Sigmoid** có thể được sử dụng để đưa ra xác suất độc lập cho mỗi nhãn (dịch vụ).
* Đầu ra của mô hình sẽ có một vector với kích thước bằng số lượng nhãn, mỗi giá trị trong vector này biểu thị xác suất cho từng nhãn.

**c. Giới thiệu multi-label ở sklearn:**

* **Scikit-Learn** hỗ trợ xử lý multi-label classification thông qua một số công cụ đặc biệt. Các công cụ này giúp sử dụng các mô hình phân loại truyền thống để giải quyết bài toán multi-label như MultiOutputClassifier, Classifier Chains, Label Powerset …
* Các phương pháp đánh giá như accuracy, F1-score, và AUC đều cần được điều chỉnh để phù hợp với bài toán multi-label.
* Ví dụ: Hamming Loss là một metric thường được sử dụng để đánh giá mô hình multi-label, đo lường tỷ lệ các nhãn bị dự đoán sai.

**6. Giới thiệu các đặc trưng có thể rút trích từ observation ở mục 4**

**Các đặc trưng có thể rút trích bao gồm:**

* Các thành phần thời gian (year, month, day, day\_of\_week).
* Chu kỳ (period) và dữ liệu (data): Cung cấp thông tin về mức độ hoạt động và hiệu suất của các dịch vụ tại thời điểm quan sát.

**7. Giới thiệu model**

**a. Model bạn chọn thuộc nhóm 5.b hay 5.c**

Model được sử dụng trong dự án này là RandomForest thuộc multi-label ở sklearn.

* **Lý do chọn Random Forest**:
* **Không yêu cầu nhiều tiền xử lý dữ liệu**: Random Forest hoạt động tốt với dữ liệu có các đặc trưng khác nhau, bao gồm cả số nguyên và số thực.
* **Dễ điều chỉnh và diễn giải**: Random Forest có thể dễ dàng diễn giải thông qua việc quan sát độ quan trọng của các đặc trưng.
* **Tránh overfitting**: Với sự kết hợp của nhiều cây quyết định, Random Forest giúp giảm khả năng overfitting so với việc sử dụng một cây quyết định đơn lẻ.
* **Label độc lập:** thông qua **data visualization** thấy rằng sự tương quan giữa các label không cao.

**b. Input, output của model**

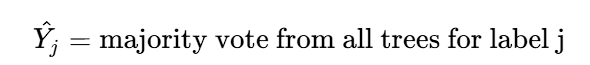
* Input: Các đặc trưng rút trích từ thời điểm T.
* Output: Các nhãn dịch vụ có thể bị lỗi tại T+deltatime.

**c. Phương trình thể hiện mối liên hệ giữa input và output.**

Random Forest là một thuật toán dựa trên cây quyết định nên không có một phương trình đơn giản giống như các mô hình tuyến tính hay mạng nơ-ron. Thay vào đó, nó hoạt động dựa trên sự kết hợp của nhiều cây quyết định độc lập.

Mối liên hệ giữa **đầu vào** (input) và **đầu ra** (output) được biểu diễn bằng cách tổng hợp đầu ra từ tất cả các cây trong rừng:

* Giả sử chúng ta có N cây, mỗi cây sẽ tạo ra một dự đoán cho đầu vào.
* **Đầu ra cuối cùng** của mô hình là **kết quả bỏ phiếu** của tất cả các cây:



**d. Kết quả của quá trình huấn luyện**

* Kết quả của quá trình huấn luyện là một **mô hình Random Forest** với các cây đã được huấn luyện để dự đoán lỗi của các dịch vụ.
* Mô hình sẽ học được các quy luật từ **quan sát thời gian trước đó (**trước thời điểm T**)** để dự đoán **khả năng gặp lỗi tại thời điểm T+deltatime.**

**e. Ta có thể dùng những tham số điều khiển nào**

* Ta có thể sử dụng các siêu tham số trong model RandomForest như n\_estimators (Số lượng cây trong model), min\_samples\_split (số lượng mẫu tối thiểu chia cho mỗi nút)…

**8. Giới thiệu metric đánh giá cho bài toán multi-label**

Để đánh giá mô hình multi-label classification, cần sử dụng các metric đặc biệt để đánh giá mức độ hiệu quả trong việc dự đoán nhiều nhãn cùng lúc. Dưới đây là các metric phổ biến và chi tiết cho bài toán multi-label.

1. **Accuracy (ACC)**

Subset Accuracy (hay còn gọi là Exact Match Ratio) đo lường tỷ lệ các mẫu mà tất cả các nhãn được dự đoán chính xác so với nhãn thực tế. Đây là một cách đánh giá rất "nghiêm khắc", vì toàn bộ các nhãn của một mẫu phải được dự đoán đúng mới được tính là chính xác.

A black and white text

Description automatically generated

1. **F1 Score**

* **F1 Score** đo lường sự cân bằng giữa **Precision** (độ chính xác) và **Recall** (độ phủ) trong việc dự đoán từng nhãn.

A screenshot of a math problem

Description automatically generated

1. **AUC (Area Under Curve)**

**AUC-ROC** là một metric phổ biến dùng để đánh giá khả năng của mô hình trong việc **phân biệt** giữa các nhãn **dương** (positive) và **âm** (negative). **AUC** là diện tích dưới đường cong **ROC**.

AUC thể hiện xác suất mà mô hình có thể phân biệt chính xác giữa một mẫu dương và một mẫu âm. Giá trị AUC dao động từ 0.5 (dự đoán ngẫu nhiên) đến 1.0 (hoàn hảo).

**9. Thí nghiệm**

**a. Chạy thử nghiệm với các hyperparameter tuning**

* Thực hiện tinh chỉnh các siêu tham số trong model RandomForest bằng cách sử dụng GridSearchCV

**b. Report kết quả với ACC, F1score, AUC:**

* Kết quả các chỉ số sẽ được so sánh để lựa chọn mô hình có hiệu suất cao nhất.

**10. Document**

|  |  |
| --- | --- |
| **Code** | **Ý nghĩa** |
| **import matplotlib.pyplot as plt**  **import seaborn as sns**  **# Pie chart for class distribution**  **def Column\_chart(y):**  **y.sum().plot(kind="bar")**  **plt.ylabel("Numbers")**  **plt.show()** | **Vẽ biểu đồ cột thể hiện sự phân tán của labels**  **y: labels** |
| **def balance\_data\_undersampling(df, labels):**  **# labels là dataframe gồm các cột Outcome**  **# Đếm số lượng mẫu của mỗi lớp**  **label\_counts = labels.sum(axis=0)**  **# Đặt ngưỡng mục tiêu để cân bằng cho tất cả các nhãn với giá trị hợp lý**  **target\_count = min(label\_counts.max(), 700)  # Đặt ngưỡng cân bằng là 300 để giảm số lượng mẫu của các nhãn lớn**  **# Tạo danh sách các DataFrames đã được xử lý cho từng nhãn (undersampling)**  **undersampled\_dfs = []**  **for col in labels.columns:**  **# Lấy tất cả các mẫu có nhãn hiện tại là 1 và thực hiện undersampling nếu cần**  **positive\_samples = df[df[col] == 1]**  **if len(positive\_samples) > target\_count:**  **positive\_samples = positive\_samples.sample(n=target\_count, random\_state=42)**  **# Thêm vào danh sách các DataFrames đã xử lý**  **undersampled\_dfs.append(positive\_samples)**  **# Kết hợp tất cả các DataFrames lại với nhau và loại bỏ các bản sao trùng lặp**  **final\_undersampled\_df = pd.concat(undersampled\_dfs).drop\_duplicates().reset\_index(drop=True)**  **return  final\_undersampled\_df** | **Undersampling dữ liệu** |
| **def change\_col\_position(df):**  **cols = df.columns.tolist()**  **new\_order = [cols[-1]] + cols[:-1]**  **df = df[new\_order]**  **return df** | **Dịch chuyển cột cuối cùng của dataframe lên đầu** |
| **def enter\_time():**  **while True:**  **start\_time = input("Nhập giá trị timestamp (định dạng: yyyy-mm-dd hh:mm:ss): ")**  **try:**  **start\_time = pd.to\_datetime(start\_time)**  **break**  **except ValueError:**  **print("Vui lòng nhập chính xác theo định dạng yêu cầu")**  **return start\_time** | **Hàm nhập thời gian T** |

**Các bước xây dựng model**

**A blue and black diagram

Description automatically generated with medium confidence**